

RDB Connect

By



Web: <http://www.prodatacomputer.com>

Introduction	4
Installation.....	6
*Configuring RDB Connect.....	11
New prompt windows added to RDBCFG.....	14
Configuring a RDB Custom Database	15
Configuring a PC Database	18
The Commands	20
<i>RDBPTF (RDB PTF Processor).....</i>	<i>20</i>
<i>RDBSEC (RDB Security Code).....</i>	<i>20</i>
<i>RDBTRACE (Set RDB Tracing).....</i>	<i>20</i>
<i>RDBFIELDS (Retrieve field information).....</i>	<i>21</i>
<i>RDBRUNSQL (Run RDB Sql Statement).....</i>	<i>22</i>
<i>RDBIMPORT (Import Remote Database)</i>	<i>22</i>
<i>RDBCFG (Remote Database Configuration)</i>	<i>23</i>
<i>RDBJARCFG (Custom Database Configuration)</i>	<i>23</i>
<i>RDBTABLES (Retrieve table list)</i>	<i>24</i>
The Functions.....	27
<i>RDB Connect (Connect to the remote server).....</i>	<i>27</i>
<i>RDBCclose (Close any open connection).....</i>	<i>28</i>
<i>RDBExec (Execute a SQL statement on the remote server).....</i>	<i>29</i>
<i>RDBPrepStmt (Create a prepared SQL statement on the remote server).....</i>	<i>30</i>
<i>RDBPrepExec (Execute a previously prepared SQL statement on the remote server).....</i>	<i>31</i>
<i>RDBFreeStmt (Free the previously executed statement)</i>	<i>32</i>
<i>RDBError (Returns the errors that occurred)</i>	<i>33</i>
<i>RDBFetchNxt (Fetch the next available record)</i>	<i>34</i>
<i>RDBFetchPrv (Fetch the previous record).....</i>	<i>35</i>
<i>RDBFetchAbs (Fetch the absolute record).....</i>	<i>36</i>
<i>RDBGetNum (Get a numeric field from a record).....</i>	<i>37</i>
<i>RDBAscNum (Get a numeric field from a record using field name).....</i>	<i>38</i>
<i>RDBGetStr (Get a character field from a record)</i>	<i>39</i>

<i>RDBAscStr (Get a character field from a record using field name)</i>	40
<i>RDBGetDate (Get a date/time/timestamp field from a record)</i>	41
<i>RDBSetNum (Set a numeric field to a parameter marker)</i>	43
<i>RDBSetStr (Set a string field to a parameter marker)</i>	44
<i>RDBSetDate (Set a timestamp field to a parameter marker)</i>	45
<i>RDBSetNull (Set a NULL value to a parameter marker)</i>	46
<i>RDBSetCommit(Set commitment control)</i>	47
<i>RDBCommit(Commit all transactions)</i>	48
<i>RDBRollback(Rollback all transactions)</i>	49
<i>RDBAddRec(Add a record to the remote database)</i>	50
<i>RDBUpdRec(Update a record in the remote database)</i>	51
<i>RBBDelRec(Delete a record in the remote database)</i>	52
<i>RDBNextSet(Move the cursor to the next result set)</i>	53
<i>RDBStoredProc (Create a SQL statement to execute a stored procedure on the remote server)</i>	54
<i>RDBRegOutput (Register an output parameter of a stored procedure)</i>	55
<i>RDBGetParmNum (Get a numeric field from a stored procedure parameter)</i>	56
<i>RDBGetParmStr (Get a character field from a stored procedure parameter)</i>	57
<i>RDBGetParmDate (Get a date/time/timestamp field from a stored procedure parameter)</i>	58
<i>*RDBCrtTable(Create a table in any Database configured in RDBCFCG)</i>	59
<i>*RDBSetIsoDate(Set ISO Date in result set)</i>	60
<i>*RDBSetIsoTime(Set ISO Time in result set)</i>	61
<i>*RDBSetCharStr(Set characters stream after executing RDBPrepStmnt)</i>	62
<i>*RDBGetCharStr(Get characters stream after executing RDBPrepExec)</i>	63
<i>*RDBGetIsoTime(Get ISO Time in result set)</i>	64
<i>*RDBGetIsoDate(Get ISO Date in result set)</i>	65
<i>*RDBAscIsoTime(Get ISO Time in result set by Field Name)</i>	65
<i>*RDBAscIsoDate(Get ISO Date in result set by Field Name)</i>	67
<i>RDBSetInt (Set a Integer field to a parameter marker)</i>	68
<i>RDBSetNegInt (Set a negative Integer field to a parameter marker)</i>	69

Introduction

RDB Connect is a collection of commands and functions allowing IBM i users connection to remote databases. RDB Connect is pre-configured to access MySQL, Microsoft SQL Server, Oracle, Postgre, DB2 8.1 and above, and DB2 for the IBM i. Other databases can be configured manually (see RDBJARCFG). An ODBC connection is also available via an included PC component.

RDB Connect will run on IBM i with an operating system version of V5R4M0 and above. It requires an IP connection to the remote server that is running the database to access.

This document will cover the usage of RDB Connect functions and the commands that are shipped with the software. Service program function prototypes and example usage are available in the source file **RDB40/RDBSRC**.

After installing RDB Connect, a subsystem called **RDBSBS** will be created in the RDB40 library. This subsystem must be active to use RDB Connect. The subsystem will contain a job called **RDB CONNECT** jobs that handle the processing of the requests.

Technical support is available M-T 8:00 am - 5:00 pm CST. Friday 8:00 am – 4:00 pm CST (except holidays)

Email – help@prodatacomputer.com

Phone – 1.800.228.6318 option 2

NOTE: Before installing RDBConnect 4, check if Java 1.5 or above is installed and running on your System for better performance and JDBC driver compatibility.

Installation

Step #1

A splash screen will appear and a series of notices informing you of the process being performed. After which the following screen should appear. Click the **Next** button to continue the installation process.

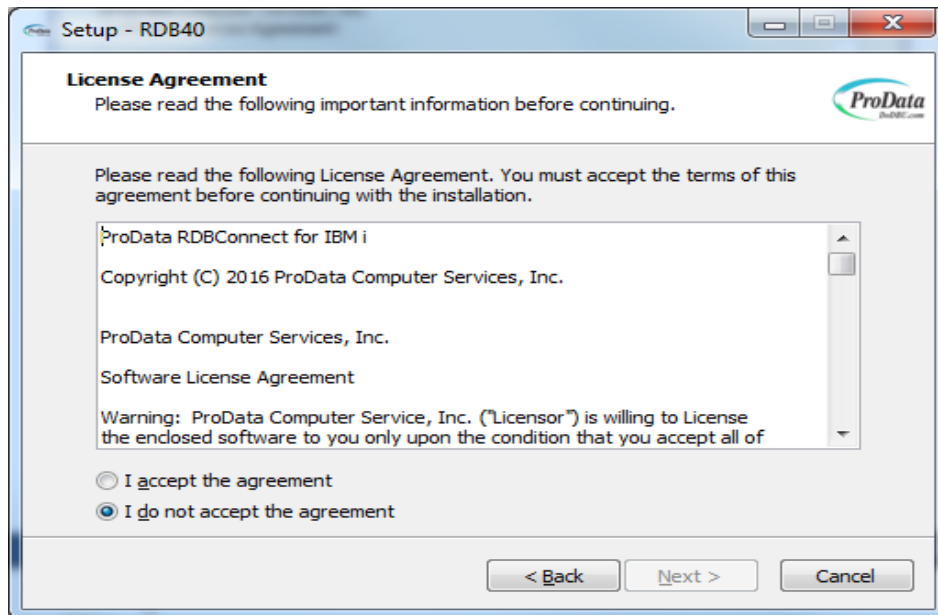


Step #2

The installation process requires a connection being established to your iSeries (AS/400) host computer. The following notice may appear informing you for the need of a connection. Once you have verified the connection to your iSeries (AS/400) host.

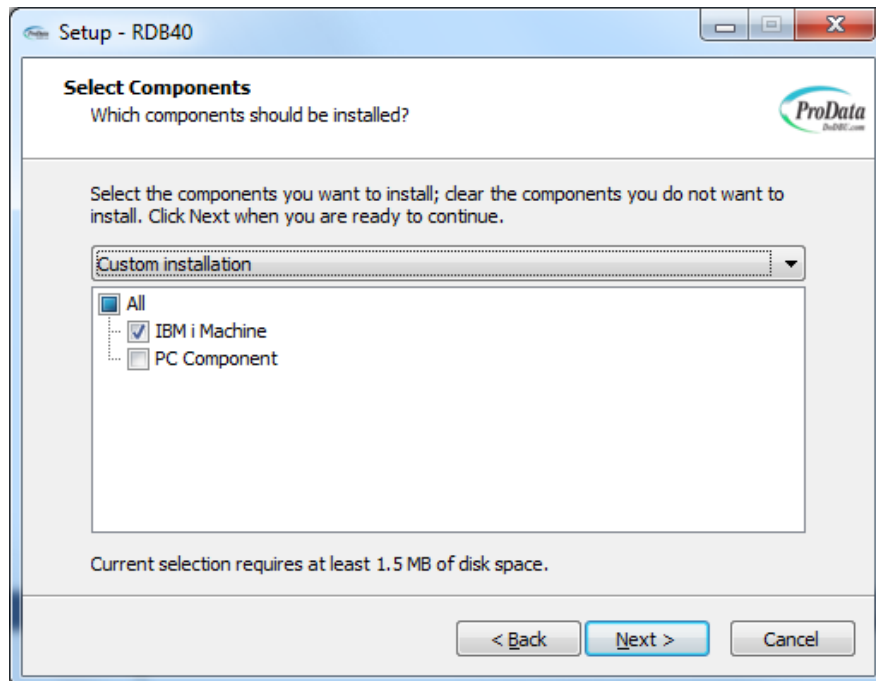
You must have *IOSYSCFG authority. Once you have verified your authority, click the OK button to continue the installation process.

Step #3 Please read the *License Agreement* and upon accepting the agreement, click the **Next** button to continue the installation process.



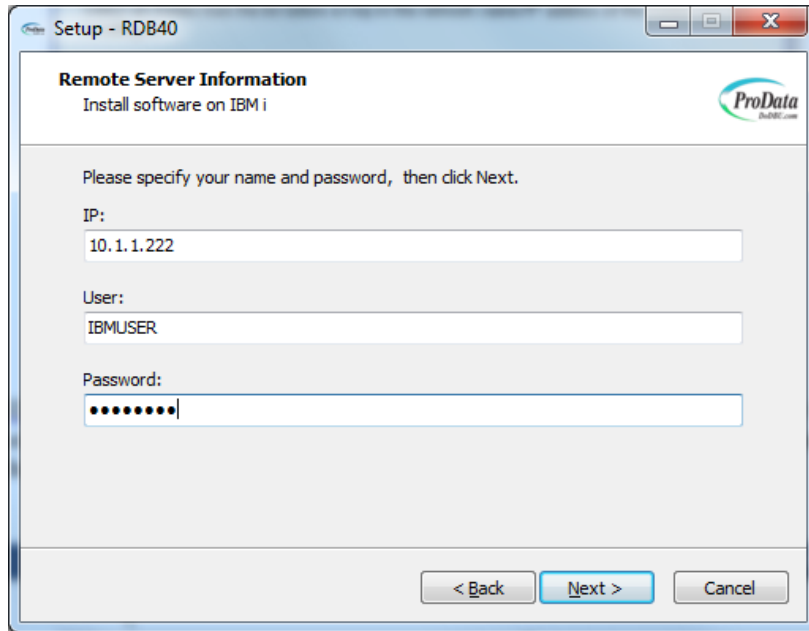
Step #4

This screen gives you the option to select which components you would like to install. You must select IBM i Machine to complete the installation.



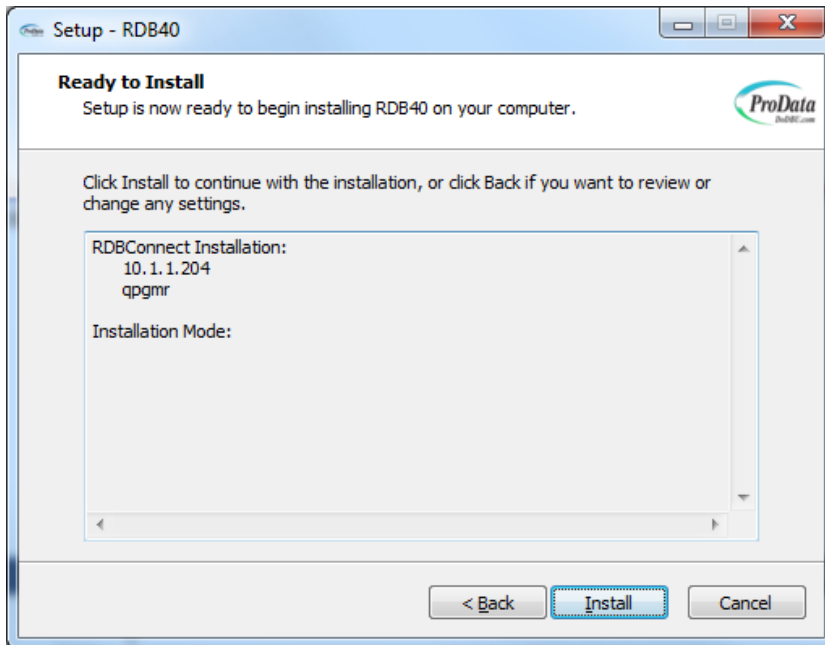
Step #5

Type the **IP Address, User and Password** in the space provided of the iSeries (AS/400) host computer. Once you have completed the iSeries (AS/400) host computer selection, click the **Next** button to continue.



Step #6

Click Next to install RDBConnect on the iSeries(AS/400) host computer.

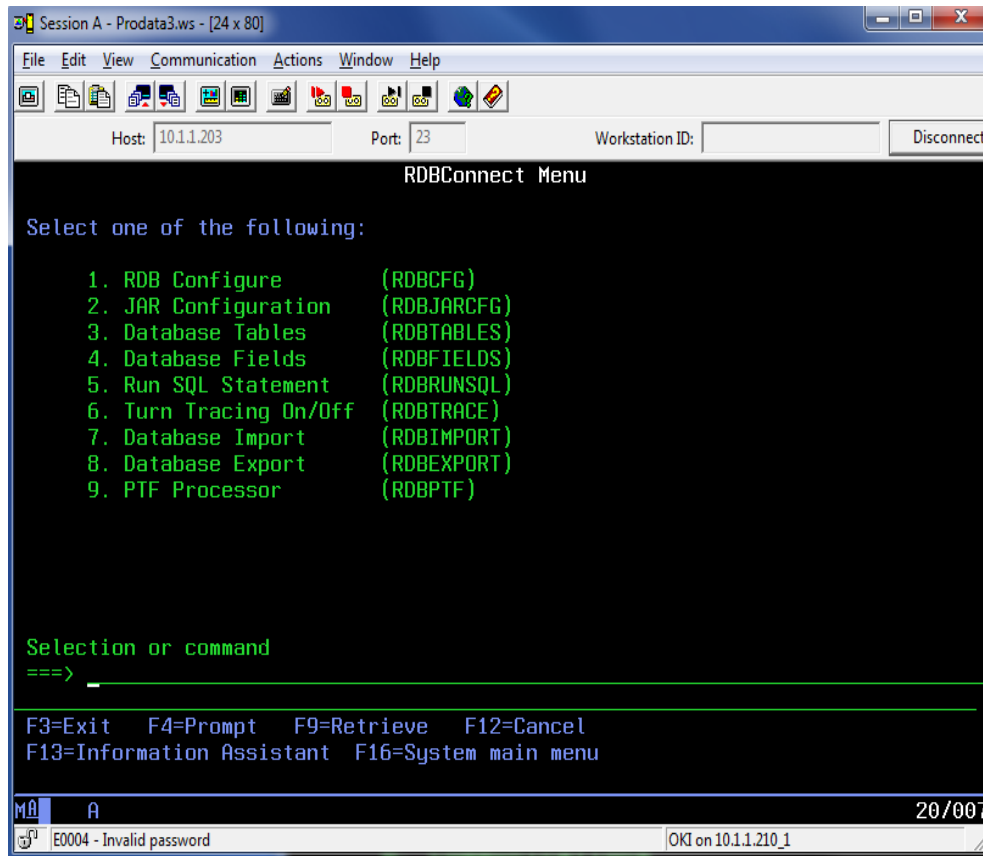


Step #7 The last screen that should appear is the **Completion** screen. Press the **Finish** button to complete the installation process.



*RDBConnect Menu

To start the RDB Connect menu, from a IBM I command line execute the command **ADDLIBLE RDB40** . Execute the command **RDBMNU**. Type an option on the command line to execute the program.



*Configuring RDB Connect

To start configuring RDB Connect, from a IBM i command line execute the command **RDBCFCG** or option 1 from the RDB Connect Menu. This will take you to the **Manage Remote Systems** screen.

```
Manage Remote Systems

Job Status.....:  Active          Java Version...: Press F22 to display
Turn Tracing On....:  Y
Server port number.:  9082
Log File Size...:   1 MB
Log Size Max...:    64 MB

Type Option, Press Enter
1=Update 4=Delete 5=Display 7=Edit Authority 8=Tables 9=Test Connection
-----
Opt RDB Name  DB Type  Remote Connection Description
-----
 1  ALEXMSSQL  MSSQL    MSSQL on my PC
 2  CSVTEST   ODBC     CSV File Test
 3  CUBRIDDEMO CUBRID   Cubrid Demo
 4  DEMOTEST  ODBC     RDB Demo Test
 5  MARIAMARIA MARIADB  Maria DB
 6  MIKETST   ODBC     install test
 7  ORACLE    ORACLE   Oracle 12c on Virtual Machine
 8  RDBACCESS ODBC     MS ACCESS DB
 9  RDBDB2    DB2      DB2 Test
10  RDBFIREB  FIREBIRD FireBird Database

F1=Hlp F3=Exit F6=Add F7=Logs F8=IFS F20=End Job F21=Clr Logs F22=JVM

M  B                                     08/038
```

Maximum Connections – This is the maximum number of threads the **RDB CONNECT** job will create. The default number of jobs is 50.

Turn Tracing On – Default is N. Set to Y when troubleshooting with ProData Tech Support. This value can be changed at any time from this program. The command **RDBTRACE** can also be used for this function.

Server port number – This is the IP port that the **RDB CONNECT** job will use to answer requests for RDB transactions. This port must be available. 9082 is the default. Press **F6** to add a remote connection.

***Job Status** - This indicates the RDBConnect job status in subsystem RDBSBS

***Log File Size** – This is the actual size of the log tracing file in IFS

***Log Size Max** – This is the maximum file size before a warning message is sent to the operator if the warning flag is set to 'Y'.

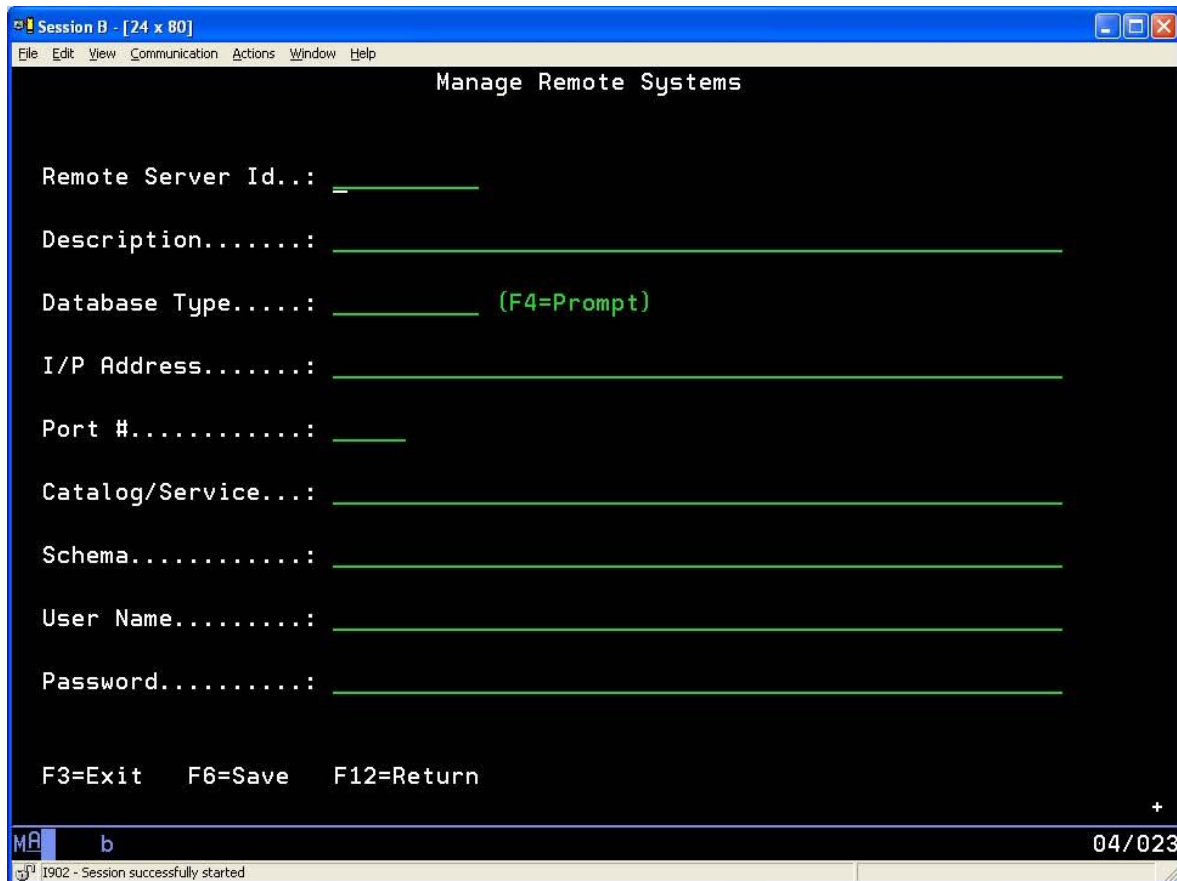
***Function Keys** – This will explain the addition of new function keys.

F7=Trace Logs – This will display the trace logs stored in IFS.

F8=IFS – This will take you to the IFS directory

F20=End Job – This will end the subsystem job RDBConnect in RDBSBS

F21= Clear Logs – This will clear the IFS trace logs.



Enter the information for the remote database.

Remote Server ID – This is the name that will be used when referencing this database connection.

Description – The description for this remote connection.

Database Type – The type of database being configured. Preconfigured databases:

MYSQL – www.mysql.org (an open source database)

MSSQL – Microsoft SQL Server 2000 and above (JTDS driver)

MSSQL2 – Microsoft SQL Server 2000 and above (Microsoft driver). Requires i5/OS V5R4 and above using JRE 6 and above.

ORACLE – Oracle server 9i and above

POSTGRE – www.postgresql.org (an open source database)

DB2 – IBM DB2 8 and above (not OS/400 or i5/OS)

DB2I – IBM DB2 for OS/400 and i5/OS

FIREBIRD – www.firebirdsql.com (an open source database)

ODBC – A PC based database using a System DSN. Must have RDB PC module loaded.

I/P Address – The I/P address used to access the remote database. This can also be an entry from the host table.

Port # – The port number to be used with the above I/P address to connect to the remote

database. If using the default port for the specified database type, this can be left blank. When in doubt, specify the port number.

Catalog/Service – Some databases require a Catalog or Service to be specified on the connection. If one is required for the database specified, enter it here.

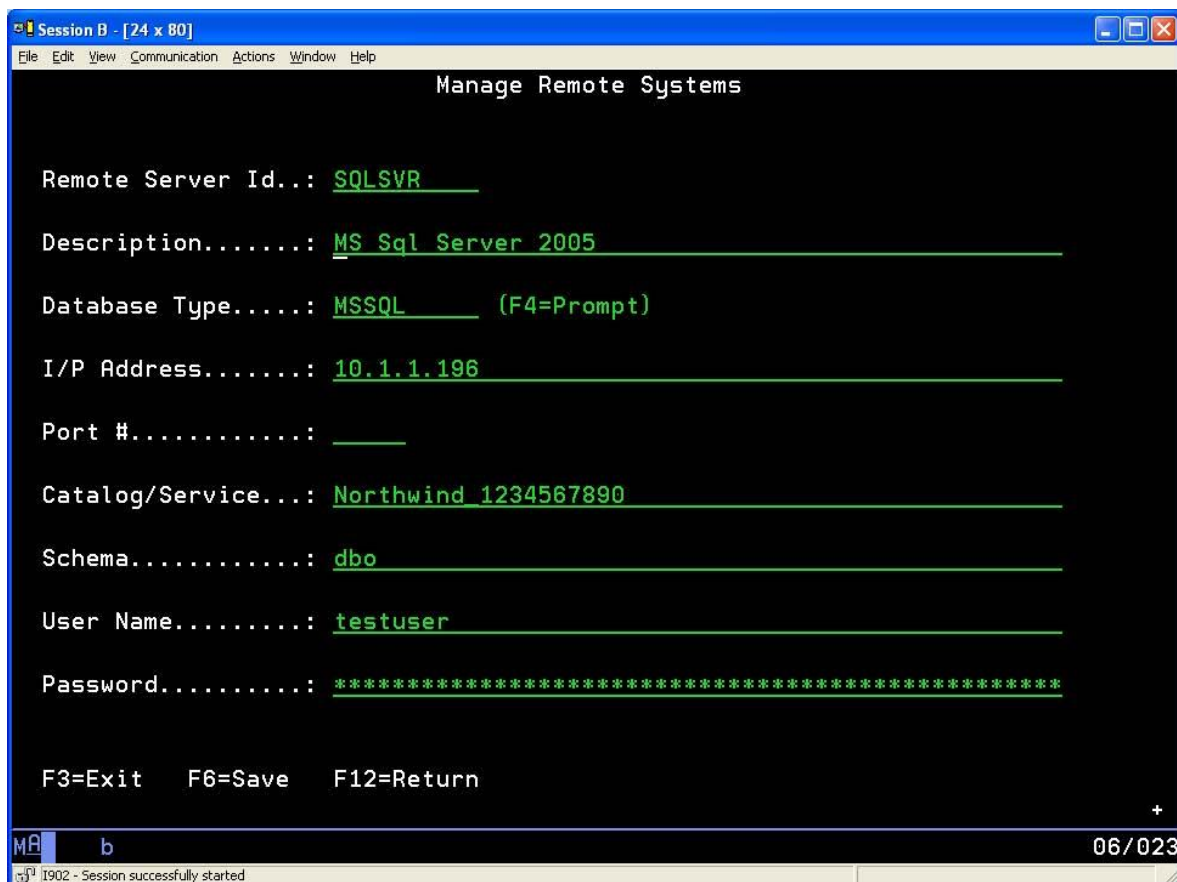
Schema – The schema for the remote database.

User Name – The user to be used when connecting. This user id will control the authority on the remote database when using this connection. This user name must be configured on the remote database. ** For Microsoft SQL Server using Windows (NTLM) authentication instead of the usual SQL Server authentication, the user name should be in the format of domain/user. This allows non-Windows clients to log in to servers which are only configured to accept Windows authentication.

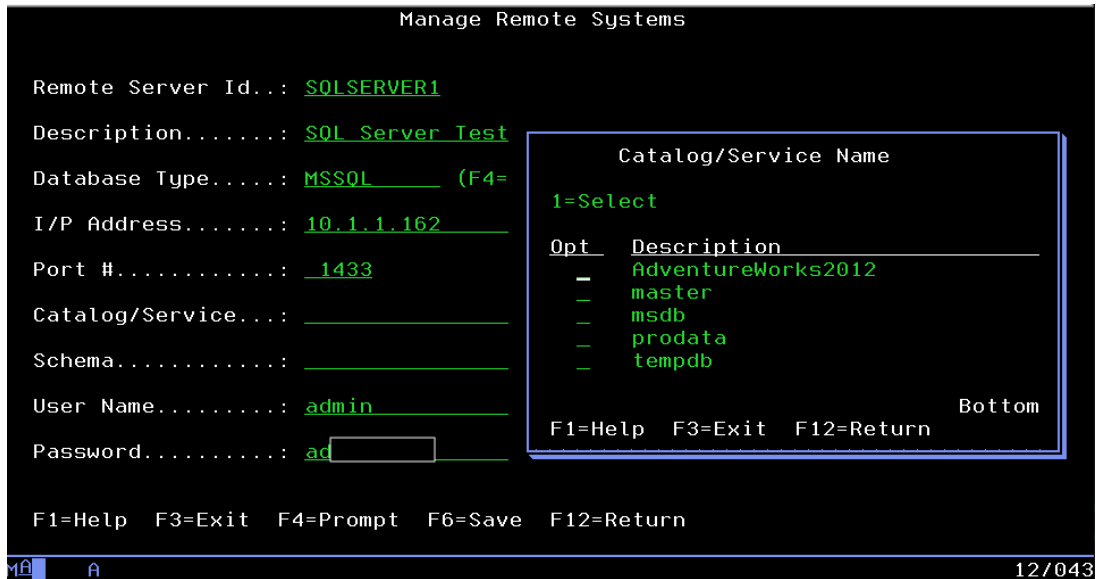
Password – The password for the above entered user name.

To save the information, press **F6**. The information entered will be encrypted 256 bit and stored in a IBM i object. The object will be named the same as the **Remote Database ID** and be placed in the RDB library. IBM i object level authority can be used, with this object, to add another level of security to the remote connection.

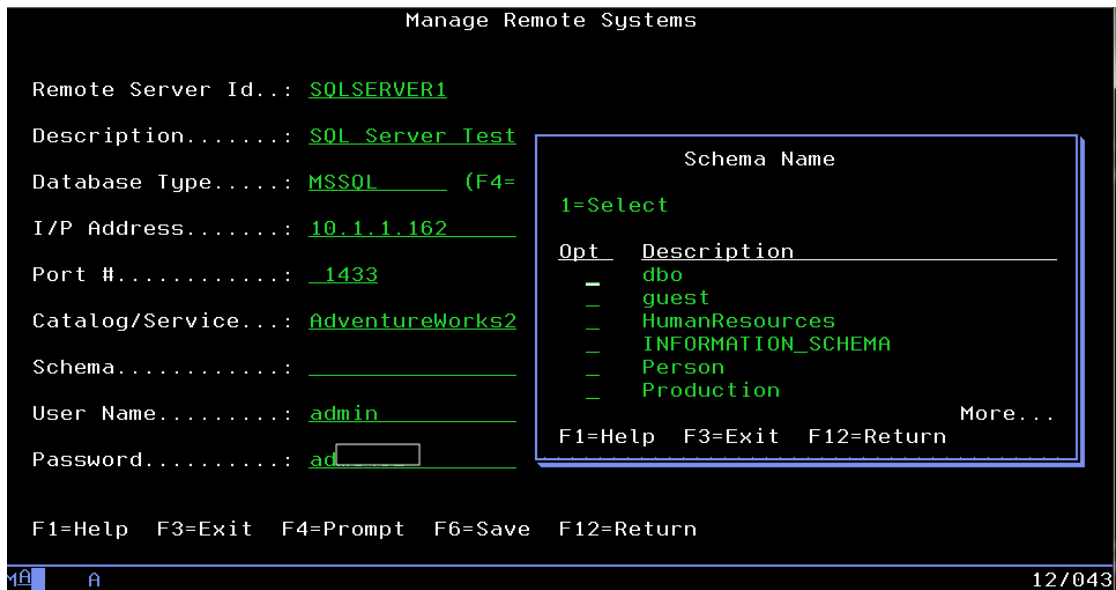
The example below is for a SQL Server configuration.



New prompt windows added to RDBCFG



New prompt field added to display all Catalog/Service Names. A User ID, Password, valid IP Address and Port must be provided for this option to work.

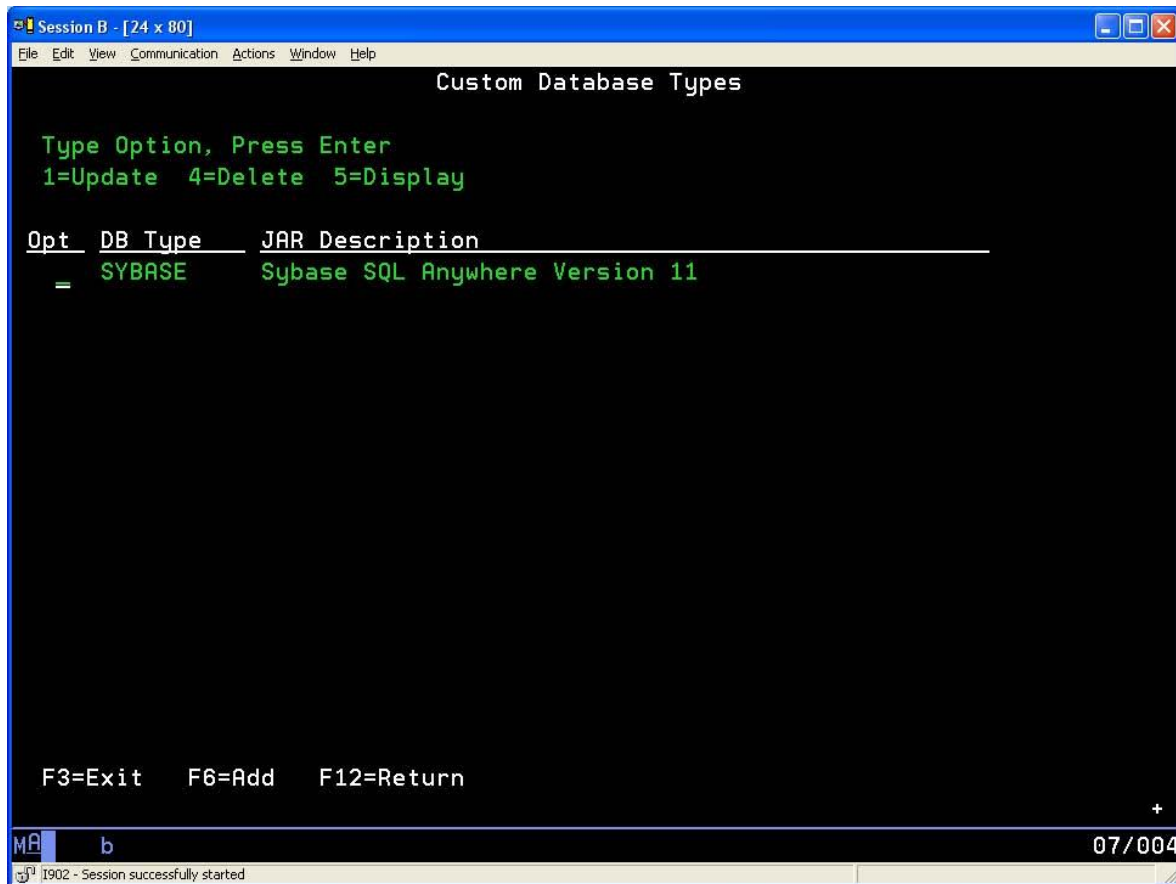


New Schema prompt – Retrieve a list of schemas. A valid Catalog/Service name must be provided for this option to work.

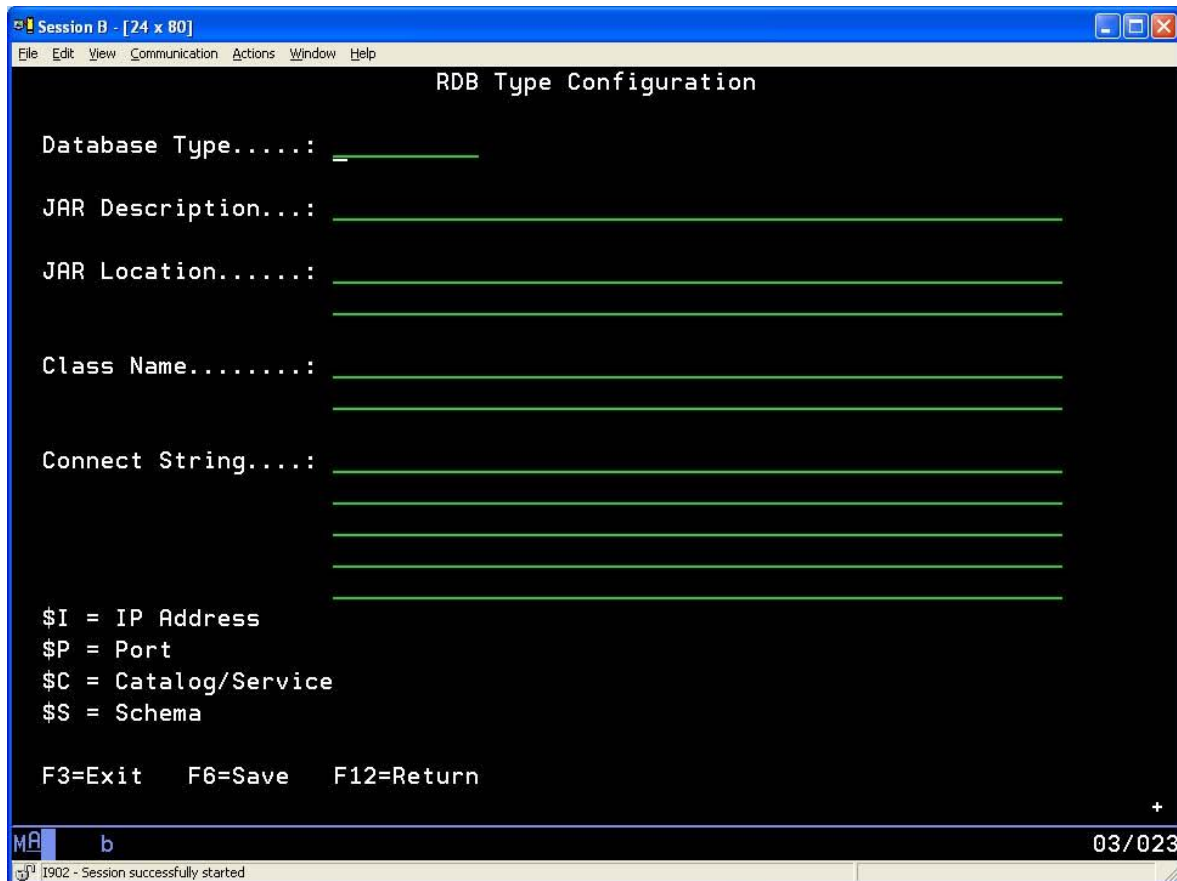
NOTE: Currently the Catalog prompt will work on the following remote databases
 MSSQL, MYSQL and POSTGRES
 The Schema prompt will work on the following remote databases
 MSSQL, ORACLE and POSTGRES

Configuring a RDB Custom Database

RDB Connect can be configured to access any database of your choosing. You must have a JDBC driver for the database and it must be in a directory on the IFS. To configure a custom database execute the command **RDBJARCFG**. This will take you to the **Custom Database Types** screen.



Press **F6** to add a custom database.



Enter the information for the custom database.

Database Type – This is the name that will be used when referencing this custom database type. It will appear in the prompt list when configuring a connection.

JAR Description – The description for this custom connection.

JAR Location – The path on the IFS for the jar file containing the JDBC driver of this database.

Class Name – The JDBC drive class name. This must be the complete jar path

Connect String – The string used to connect to the custom database. This string must contain the \$I substitution variable and optionally the \$P, \$C, and \$S.

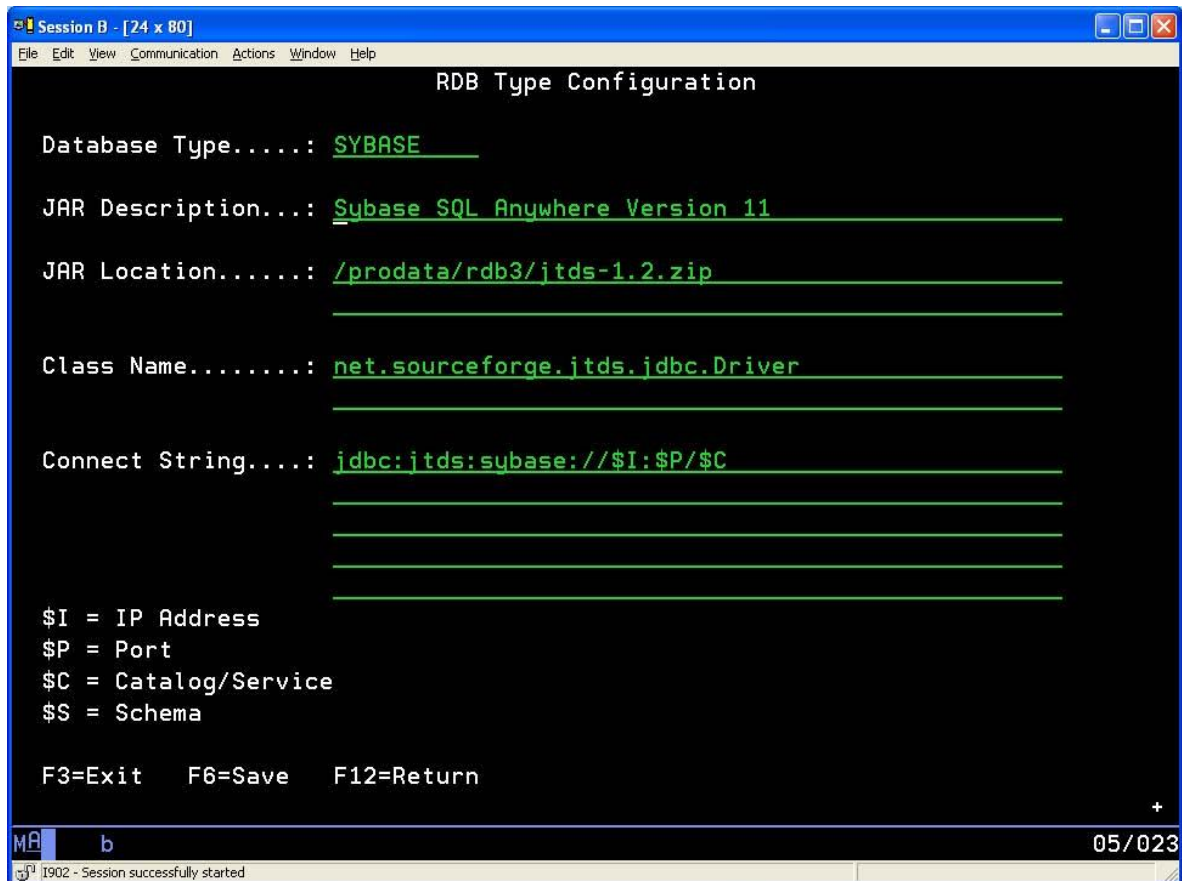
\$I – The IP address of the remote database will be placed at this point in the connection string.

\$P – The port number of the remote database will be placed at this point in the connection string.

\$C – The Catalog/Service of the remote database will be placed at this point in the connection string.

\$S – The Schema of the remote database will be placed at this point in the connection string.

The example below is for a Sybase configuration.



Configuring a PC Database

RDB Connect can be configured to access a database on a PC that does not have a JDBC driver. The database must have an available ODBC driver. This function requires two things. The RDB service running on the PC and a DSN defined for that database.

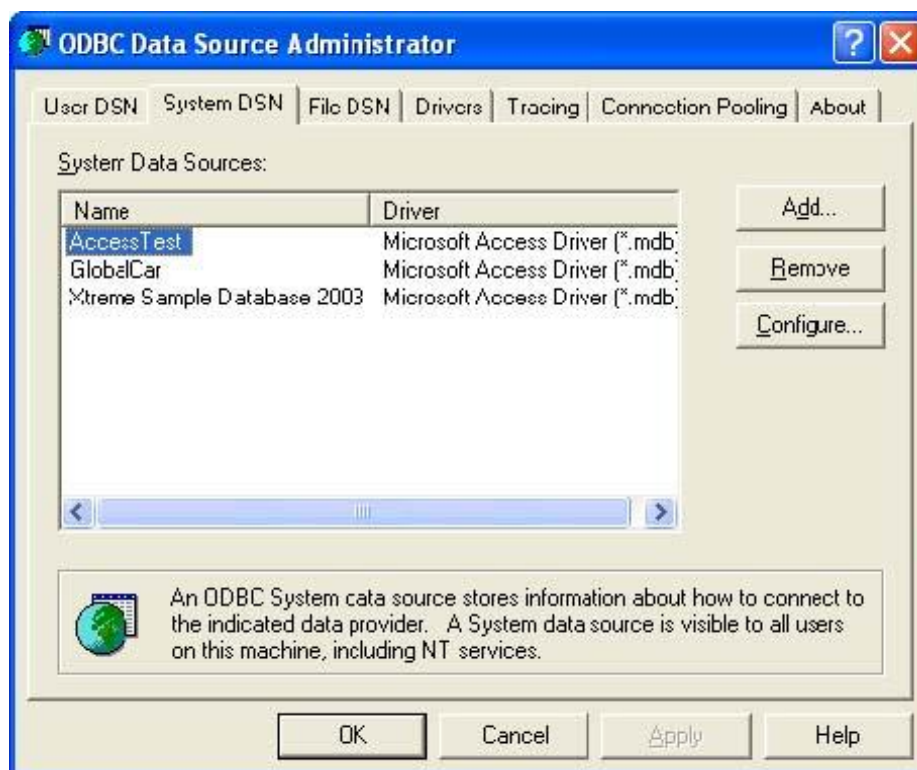
When installing the RDB software, there is a checkbox for the “PC Component”. Select this option to install the PC software on the appropriate computer. This will create a directory “Program Files/Prodata/RDB4” and place the required programs in that directory. Two batch files will also be placed in the directory. The file “RDBServiceStart.bat” must be modified. The first line in the file is the configuration command for the service. It contains the path to the JVM on your PC. This path must be correct. If it is not, please change it. Example:

```
RDBService -i -J "C:\\Program Files\\Java\\jre1.6.0_05\\bin\\client\\jvm.dll" -P 9082 -L 100 -T
```

The portion that needs to be modified is following the “-J” parameter. This is the path to the jvm.dll object. The “-P” parameter controls the port that will be used for communication to the PC. The default is 9082. The “-L” parameter controls the number of listeners that will process requests on this PC. The default is 100. The “-T” parameter controls the use of the trace log function. This is useful to troubleshoot issues. The “-i” parameter causes the service to be installed.

After the batch file has been changed to contain the correct values, it can be executed to start the service. To end the service, execute the “RDBServiceStop.bat” file.

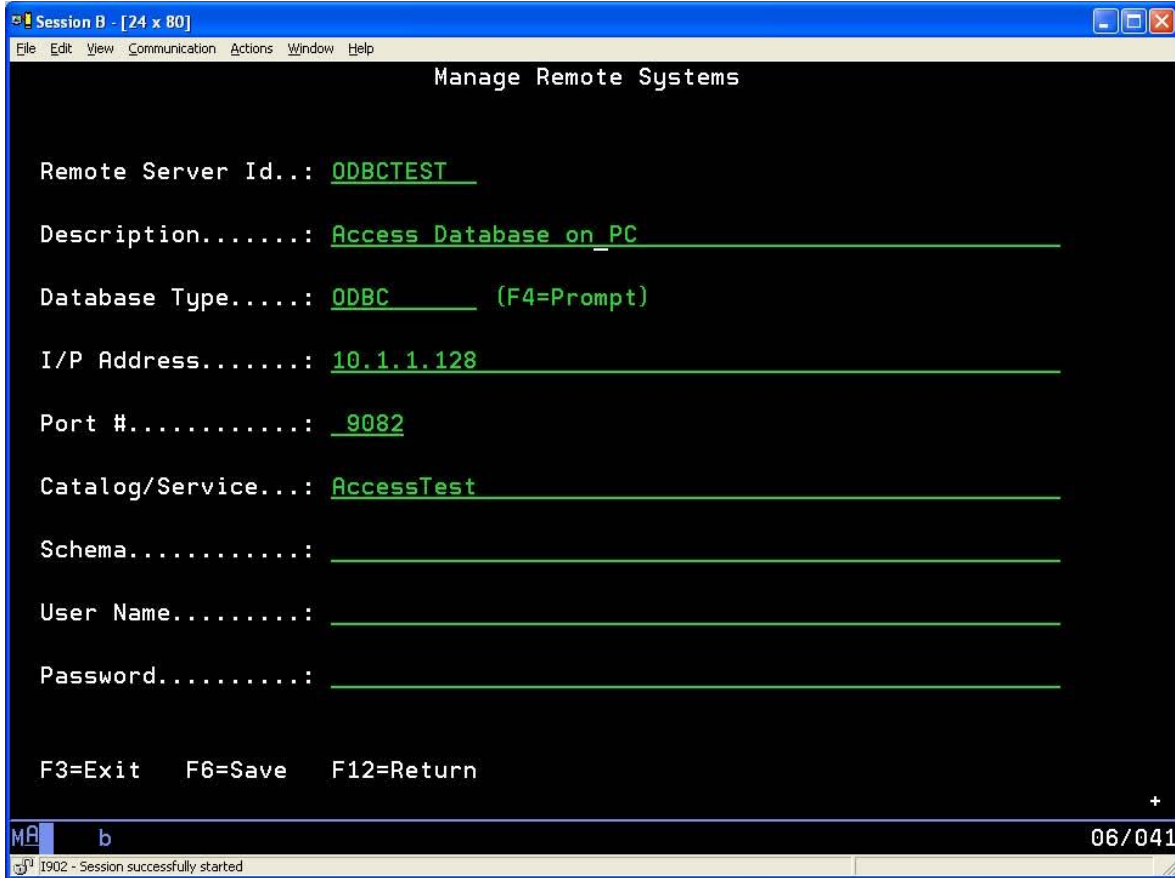
To configure a connection the PC database you must setup a DSN on the PC. This is done via the Control Panel->Administrative Tools->Data Sources (ODBC).



You can also open Starts > Type ODBC and click on the 32bit or 64bit version

The DSN needs to be defined as a “System DSN”. This will allow the RDB Service to access the definition.

On the IBM i, use the **RDBCFCG** to configure the connection to the ODBC database. The screen below shows an example ODBC configuration.



The “System DSN” that was defined on the PC is placed in the “Catalog/Service” field.

The Commands

RDBPTF (RDB PTF Processor)

The RDB PTF Processor (RDBPTF) command allows you to retrieve updates for RDB Connect. The process makes a connection to Prodata Computer Services using port **2809**. If problems occur during the running of this command, verify your firewall is not blocking this transaction. This function will only retrieve the programs that have been updated since your last update.

Parameters

Keyword	Description	Default	Notes
LIB	The library to receive the updated programs	RDB40	This must be the library that currently contains RDB Connect

RDBSEC (RDB Security Code)

The RDB Security Code (RDBSEC) command provides an interface to enter the permanent and temporary access codes for RDB Connect.

Parameters

N/A

RDBTRACE (Set RDB Tracing)

The Set RDB Tracing (RDBTRACE) command turns the logging process on and off in the RDB CONNECT job. This is the same function as the “Turn Tracing On” in the RDBCFG screen. The flag on the RDBCFG screen is used at startup of the RDB CONNECT job. This command can be used anytime the RDB CONNECT job is running.

Parameters

Keyword	Description	Default	Notes
TRACE	Turn tracing on or off.	*ON	Valid values are *ON or *OFF
Log File Size	Set Max log file size	Number	Number of MegaBytes
Send Msg	Send Warning Msg	N	Valid values are 'Y' or 'N'
User Profile	Warning msg is sent here	Character	Valid profile ID

RDBFIELDS (Retrieve field information)

The Retrieve field information (RDBFIELDS) command provides the field definitions from the remote database. The definitions are based on the select statement provided.

Parameters

Keyword	Description	Default	Notes
STM	The SQL statement that will be used to retrieve the field listing.		Only SELECT statements are allowed.
SERVER	The remote server id that was created using RDBCFCG .		This must be a valid connection. You need to be authorized to use the connection object.
USER	The user id used in the connection process	*CONFIG	The default will get the user from the RDBCFCG of the selected server.
PASSWORD	The password used in the connection process	*CONFIG	The default will get the password from the RDBCFCG of the selected server.

Example

The following command might generate a screen similar to the one below.

```
RDBFIELDS STM('select * from [dbo].[Orders]') SERVER(SQLSVR)
```

```

Remote field descriptions

Remote Server.....:  SQLSVR
SQL Statement.....:  select * from [dbo].[Orders]

  Remote Field Name  Field Type      Usage  Null  Type  Length  Dec
-----
OrderID             int identity    I      N     S     10      0
CustomerID          nchar           B      Y     A      5
EmployeeID          int             B      Y     S     10      0
OrderDate           datetime        B      Y     Z     26
RequiredDate       datetime        B      Y     Z     26
ShippedDate         datetime        B      Y     Z     26
ShipVia             int             B      Y     S     10      0
Freight             money           B      Y     S     19      4
ShipName            nvarchar        B      Y     A     40
ShipAddress         nvarchar        B      Y     A     60
ShipCity            nvarchar        B      Y     A     15
ShipRegion          nvarchar        B      Y     A     15
ShipPostalCode     nvarchar        B      Y     A     10

F3=Exit
  
```

RDBRUNSQL (Run RDB Sql Statement)

The Run RDB Sql Statement (RDBRUNSQL) command provides an interface to execute commands on the remote database. A **SELECT** will run, however it will not generate any output.

Parameters

Keyword	Description	Default	Notes
STM	The SQL statement to process.		All statements except SELECT are allowed.
SERVER	The remote server id that was created using RDBCFCG .		This must be a valid connection. You need to be authorized to use the connection object.
USER	The user id used in the connection process	*CONFIG	The default will get the user from the RDBCFCG of the selected server.
PASSWORD	The password used in the connection process	*CONFIG	The default will get the password from the RDBCFCG of the selected server.

Example

```
RDBRUNSQL STM('drop [dbo].[Orders]') SERVER(SQLSVR)
```

RDBIMPORT (Import Remote Database)

The Import Remote Database (RDBIMPORT) command provides an interface to execute commands on the remote database and return the results to a local database file. The statement must be a **SELECT** statement.

Parameters

Keyword	Description	Default	Notes
STM	The SQL statement that the field listing will be based.		Only SELECT statements are allowed.
SERVER	The remote server id that was created using RDBCFCG .		This must be a valid connection that you are authorized.
USER	The user id used in the connection process	*CONFIG	The default will get the user from the RDBCFCG of the selected server.
PASSWORD	The password used in the connection process	*CONFIG	The default will get the password from the RDBCFCG of the selected server.

CRTADD	Create the local file	*YES	If *YES is specified, the local file cannot exist. If *NO is specified, the local file must exist.
OUTFILE	The name of the local IBM i file.		The file that will contain the results of the select.

Example

```
RDBIMPORT STM('select * from [dbo].[Orders]') SERVER (SQLSVR)
CRTADD (*YES) OUTFILE (MYLIB/ORDERS)
```

The above statement will create a file called ORDERS in the library MYLIB and write the selected records from the remote database to the file.

NOTE: when using MSSQL2 connection the max allowed combined characters for import to IBM i is 32765.

RDBCFCG (Remote Database Configuration)

The Remote Database Configuration (RDBCFCG) command provides an interface to configure the remote database connections. The definitions are created as objects in the RDB40 library. IBM i security can be applied to these objects to better secure your remote connections.

For usage of this command, see “Configuring RDB Connect”.

Parameters

N/A

RDBJARCFG (Custom Database Configuration)

The Custom Database Configuration (RDBJARCFG) command provides an interface to configure any database that has a JDBC driver.

For usage of this command, see “Configuring a RDB Custom Database”.

Parameters

N/A

RDBTABLES (Retrieve table list)

The Retrieve table list (RDBTABLES) command provides a list of available tables on the remote server. The list is based on the table parameter.

Parameters

Keyword	Description	Default	Notes
TABLE	The subset of tables to be listed.	*ALL	The % symbol is used as a wildcard in this parameter. Example: For a list of all tables beginning with “f”, the parameter would be specified as ‘f%’.
SERVER	The remote server id that was created using RDBCFCG .		This must be a valid connection. You need to be authorized to use the connection object.
USER	The user id used in the connection process	*CONFIG	The default will get the user from the RDBCFCG of the selected server.
PASSWORD	The password used in the connection process	*CONFIG	The default will get the password from the RDBCFCG of the selected server.

* Option

```

Session A - Prodata3.ws - [24 x 80]
File Edit View Communication Actions Window Help
Host: 10.1.1.203 Port: 23 Workstation ID: Disconnect
Remote Table List
Remote Server.....: RDBORACLE_ (F4=Prompt)
Table Generic.....: %
5=Display
Type Option, Press Enter
Opt Table Name
-
- OUT
- ACTSECT
- AUDIT_TRAIL$
- CAPWGTT
- CEMLSAV
- CITIES
- CODE$
- CODE_T$
- CODE_T$ TEMP
- COMMAND_RULE$
- CONFIG$
- CST100
- DATFMT
-
F1=Help F3=Exit F4=Prompt F12=Return
+
MA A 03/025
E0004 - Invalid password OKI on 10.1.1.210_1

```


*RDBExport (Export to Remote Database)

The Export Remote Database (RDBEXPORT) command provides an interface to execute commands on the remote database and return the results to a local database file.

Parameters

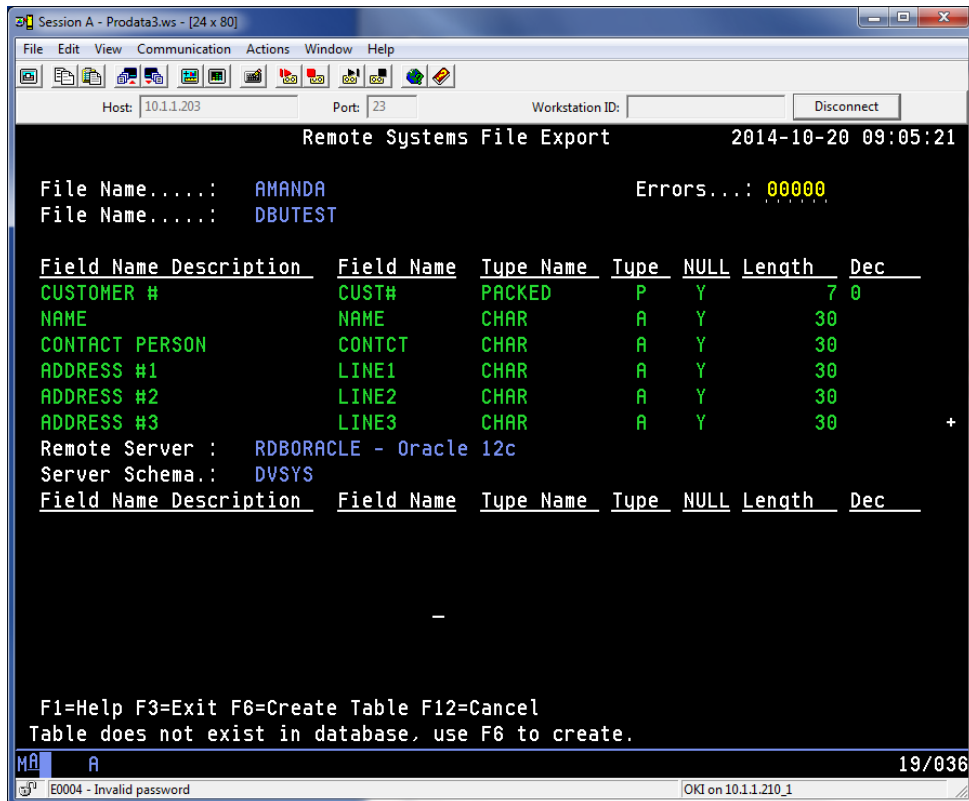
Keyword	Description	Default	Notes
REMOTE DB SQL STATEMENT	Use an SQL Select statement for all fields or individual fields.		Must specify a schema where the table resides. Select statement can include WHERE, ORDER BY and HAVING.
AS400 EXPORT FILE	Export File Name in iSeries		Specify a valid File Name
LIBRARY	LIBRARY NAME where object resides		File and Library must be correct in order to run.
SERVER	Remote Server Name specified in RDBC G command.		The default will get the user from the RDBCFCG of the selected server.
USER	The user id used in the connection process	*CONFIG	The default will get the user from the RDBCFCG of the selected server.
PASSWORD	The password used in the connection process	*CONFIG	The default will get the password from the RDBCFCG of the selected server.

```

Session A - Prodata3.ws - [24 x 80]
File Edit View Communication Actions Window Help
Host: 10.1.1.203 Port: 23 Workstation ID: Disconnect
Export To Remote Database (RDBEXPORT)
Type choices, press Enter.
Remote DB SQL Statement . . . . . select * from dvsys.amanda
AS400 Export File . . . . . amanda Name
AS400 Library . . . . . dbutest Name, *CURLIB
Remote Server Id . . . . . rdboracle Character value
User Id . . . . . *CONFIG
Password . . . . . *CONFIG
Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys
MA A 14/046
E0004 - Invalid password |OKI on 10.1.1.210_1

```

Example



RDBExport screen – Contains all the field names, field types, null allowed, length and decimal.

The top part of the screen pertains to the IBMi Database and the bottom part pertains to the remote database if a connection is established.

The errors counter is a warning to identify when the field types and lengths do not match.

The Functions

RDB Connect provides you with a service program to access your remote databases. The service program is called RDB2000 in the library RDB40. An example program and the prototypes for the supplied functions can be found in RBD40/RDBSRC. A binding directory called RDB2000 is supplied with RDB Connect to assist in the compiling of your programs. It can be found in the library RDB40.

RDB Connect (Connect to the remote server)

Purpose

RDB Connect sends the connection information to the remote server and returns an ID to be used in future transactions for this database. The returned ID is valid until it is closed.

Syntax

ID = RDB Connect(RemoteId: {user}: {password}:{port})

Function Arguments

Data Type	Argument	Use	Description
Char(10)	Remote ID	Input	The RemoteID parameter is the name that was used when creating the configuration of the remote server. See "Configuring RDB Connect".
Char(20)	User	Input(Optional)	The user parameter is used during the connection process to validate the connection to the database. If it is not specified, the user from the configuration will be used.
Char(20)	Password	Input(Optional)	The password parameter is used during the connection process to validate the connection to the database. If it is not specified, the password from the configuration will be used.
Signed(4)	Port	Input(Optional)	The port number the RDB server is listening on. This is only used when multiple RDB servers are being ran at the same time. Omitting this parameter causes the connect process to use the port number from the RDBCFG screen.
Int(10)	ID	Output	An ID is returned - will be used throughout the process to maintain the connection. To connect to a database multiple times or to multiple databases, use multiple IDs. A non-negative number signifies a valid connection.

Examples

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
*
* Connect to the remote system using the Id created in the RDB
* configuration screen (RDBCFCG)
C           Eval           Id = RDB Connect('SQLSVR')
```

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
*
* Connect to the remote system using the Id created in the RDB
* configuration screen (RDBCFCG) with a user and password
C           Eval           Id = RDB Connect('SQLSVR': : 'me': 'mypass')
```

RDBCclose (Close any open connection)

Purpose

RDBCclose closes any open connection.

Syntax

RDBCclose(Id)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	Connection ID	Input	The Connection ID parameter previously given when RDBConnect was used.

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
*
* Close the connection.
C           Callp           RdbClose (Id)
```

RDBExec (Execute a SQL statement on the remote server)

Purpose

RDBExec directly executes the specified SQL statement on the remote server. Any valid SQL statement can be executed. The syntax for the statement must be valid on the remote server.

RDB Connect() must be called before calling this function.

If a previous statement has been executed for this connection, RDBFreeStmt() must be called to close the cursor, before calling RDBExec().

Syntax

rc = RDBExec(ID: Statement: Update)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(32767) Varying	Statement	Input	The statement to be processed on the remote server.
Boolean	Update	Input (Optional)	Is the statement updatable. Valid values: *OFF – Statement is read only *ON – Statement is updatable (default)
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CL0N01Factor1++++++Opcode&ExtExtended-factor2++++++
*
*   After the connecton is established, prepare a select statement.
C           Eval      rc = RdbExec (Id:
C           \Select * from [dbo].[Orders]':*OFF)
```

RDBPrepStmt (Create a prepared SQL statement on the remote server)

Purpose

RDBPrepStmt sends the SQL statement to the remote server to be prepared.. Any valid SQL statement can be prepared. The syntax for the statement must be valid on the remote server. The SQL statement string may contain parameter markers. A parameter marker is represented by a “?” character, and indicates a position in the statement where the value of an application variable is to be substituted, when RDBPrepExec() is called. RDBSetStr(), RDBSetDate(), RDBSetNull(), and RDBSetNum() are used to associate a application variable or constant value to each parameter marker.

RDB Connect() must be called before calling this function.

If a previous statement has been executed for this connection, RDBFreeStmt() must be called to close the cursor, before calling RDBPrepStmt ().

Syntax

rc = RDBPrepStmt (ID: Statement: Update)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(32767) Varying	Statement	Input	The statement to be prepared on the remote server.
Boolean	Update	Input (Optional)	Is the statement updatable. Valid values: *OFF – Statement is read only *ON – Statement is updatable (default)
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
*
* After the connection is established, prepare a select statement.
C      Eval rc = RdbPrepStmt(Id:
C      'Select * from [dbo].[Orders]' +
C      'where ShippedDate = ?':*OFF)
```

```
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
*
* After the connection is established, prepare a select statement.
C      Eval rc = RdbPrepStmt(Id:
C      'Select * from [dbo].[Orders]' +
C      'where ShippedDate = ?':*OFF)
```

RDBPrepExec (Execute a previously prepared SQL statement on the remote server)

Purpose

RDBPrepExec executes a statement, that was successfully prepared using RDBPrepStmt(), once or multiple times. The statement is executed using the current values of any application variables that were bound to parameter markers by RDBSetStr(), RDBSetStr(), and RDBSetStr().

Syntax

```
rc = RDBPrepExec(ID)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++
*
* After the statement is prepared, execute the prepared statement.
C          Eval rc = RdbPrepExec(Id)
```

RDBFreeStmt (Free the previously executed statement)

Purpose

RDBFreeStmt ends processing on the previously executed statement. The connection to the remote system will remain open.

Syntax

RDBFreeStmt(ID)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.

Examples

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
*
* Free the statement that was last executed.
C          Callp          RdbFreeStmt (Id)
```


RDBError (Returns the errors that occurred)

Purpose

RDBError returns the error code and error text that were generated by the last executed RDB function. The output parameters will only be generated when a negative one (-1) is returned from a function.

Syntax

RDBError(Error:ErrorText)

Function Arguments

Data Type	Argument	Use	Description
Char(7)	Error	Output	The error code that was generated by the previously executed function. Error codes can be found in the RDBMSGF message file.
Char(100)	ErrorText	Output	The additional message information for the error

Examples

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
*
* Retrieve the error that was generated.
C      Callp          RdbError(Error:ErrorText)
```

RDBFetchNxt (Fetch the next available record)

Purpose

RDBFetchNxt moves the statement cursor on the remote database SELECT to the next available record. The function is only valid when a RDBExec has been used for a SELECT statement. If the fetch fails, a negative one (-1) will be returned from the function. A zero will be returned upon successful completion.

Syntax

```
rc = RDBFetchNxt(ID)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CL0N01Factor1++++++Opcode&ExtExtended-factor2++++++
* Fetch the next record of the result set that was generated with the
* previously executed statement.
C                               Eval   rc = RdbFetchNxt(Id)
```

RDBFetchPrv (Fetch the previous record)

Purpose

RDBFetchPrv moves the statement cursor on the remote database SELECT to the previously available record. The function is only valid when a RDBExec has been used for a SELECT statement. If the fetch fails, a negative one (-1) will be returned from the function. A zero will be returned upon successful completion.

Syntax

rc = RDBFetchPrv(ID)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
* Fetch the previous record of the result set that was generated with
* the previously executed statement.
C          Eval rc = RdbFetchPrv(Id)
```

RDBFetchAbs (Fetch the absolute record)

Purpose

RDBFetchAbs moves the statement cursor on the remote database SELECT to the record requested. The function is only valid when a RDBExec has been used for a SELECT statement. If the fetch fails, a negative one (-1) will be returned from the function. A zero will be returned upon successful completion.

Syntax

rc = RDBFetchAbs(ID: RecNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	RecNum	Input	The record number in the result set that was generated by a previously executed SELECT.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++Opcode&ExtExtended-factor2+++++
* Fetch a record of the result set using the absolute position of the
* result set. This will return the 4th record of the result set.
C          Eval          rc = RdbFetchAbs(Id:4)
```

RDBGetNum (Get a numeric field from a record)

Purpose

RDBGetNum retrieves the data from a numeric field in the record of the remote database. The function is only valid when RDBExec has been used for a SELECT statement and a fetch has been used.

Syntax

number = RDBGetNum(ID: FieldNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number in the result set that was generated by a previously executed SELECT.
Number(30,9)	number	Output	The value of the field in the corresponding result set is returned. Zero is returned upon failure.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++
* Fetch the first field of the result set and return it as a number.
C           Eval           Field1 = RdbGetNum(Id:1)
```

RDBAscNum (Get a numeric field from a record using field name)

Purpose

RDBAscNum retrieves the data from a numeric field in the record of the remote database using the associated field name. The function is only valid when RDBExec has been used for a SELECT statement and a fetch has been used.

Syntax

number = RDBAscNum(ID: FieldName)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(50)	FieldName	Input	The name of the field in the result set that was generated by a previously executed SELECT.
Number(30,9)	number	Output	The value of the field in the corresponding result set is returned. Zero is returned upon failure.

Examples

```
CLON01Factor1+++++Opcode&ExtExtended-factor2+++++
* Fetch the OrderID field of the result set and return it as a number.
C Eval Field1 = RdbAscNum(Id:'OrderID')
```

RDBGetStr (Get a character field from a record)

Purpose

RDBGetStr retrieves the data from a character field in the record of the remote database. The function is only valid when RDBExec has been used for a SELECT statement and a fetch has been used.

Syntax

String = RDBGetStr(ID: FieldNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number in the result set that was generated by a previously executed SELECT.
Char(32767) Varying	String	Output	The value of the field in the corresponding result set is returned. Blank is returned upon failure.

Examples

```
CLON01Factor1+++++Opcode&ExtExtended-factor2+++++
  * Fetch the first field of the result set and return it as a character
  * field.
C      Eval      Field1 = RdbGetStr(Id:1)
```

RDBAscStr (Get a character field from a record using field name)

Purpose

RDBAscStr retrieves the data from a character field in the record of the remote database using the associated field name. The function is only valid when RDBExec has been used for a SELECT statement and a fetch has been used.

Syntax

String = RDBAscStr(ID: FieldName)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(50)	FieldName	Input	The name of the field in the result set that was generated by a previously executed SELECT.
Char(32767) Varying	String	Output	The value of the field in the corresponding result set is returned. Blank is returned upon failure.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Fetch the CompanyName field of the result set and return it as a  
* character field.  
C      Eval      Field1 = RdbAscStr(Id:'CompanyName')
```


RDBGetDate (Get a date/time/timestamp field from a record)

Purpose

RDBGetDate retrieves the data from a date/time/timestamp field in the record of the remote database. The function is only valid when RDBExec has been used for a SELECT statement and a fetch has been used.

Syntax

Timestamp = RDBGetDate(ID: FieldNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number in the result set that was generated by a previously executed SELECT.
Timestamp	Timestamp	Output	The value of the field in the corresponding result set is returned. An initialized timestamp is returned upon failure.

Examples

```
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
* Fetch the first field of the result set and return it as a timestamp
* field.
C      Eval      Field1 = RdbGetDate(Id:1)
```

RDBAscDate (Get a date/time/timestamp field from a record using field name)

Purpose

RDBAscDate retrieves the data from a date/time/timestamp field in the record of the remote database using the associated field name. The function is only valid when RDBExec has been used for a SELECT statement and a fetch has been used.

Syntax

Timestamp = RDBAscDate(ID: FieldName)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(50)	FieldName	Input	The name of the field in the result set that was generated by a previously executed SELECT.
Timestamp	Timestamp	Output	The value of the field in the corresponding result set is returned. An initialized timestamp is returned upon failure.

Examples

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
* Fetch the ShipDate field of the result set and return it as a
* timestamp field.
C      Eval      Field1 = RdbAscDate(Id:'ShipDate')
```

RDBSetNum (Set a numeric field to a parameter marker)

Purpose

RDBSetNum associates a numeric application variable or constant value to a parameter marker in an SQL statement. When the statement is executed, the content of the variable is sent to the database server.

Syntax

`rc = RDBSetNum(ID: FieldNum: Value)`

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	Parameter marker number, ordered sequentially left to right, starting at 1.
Number(30,9)	Value	Input	The value to use in the corresponding parameter marker.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
* Set the value of the second parameter as a number with a value of 5
C      Eval      rc = RdbSetNum(Id: 2: 5)
```

RDBSetStr (Set a string field to a parameter marker)

Purpose

RDBSetStr associates a character application variable or constant value to a parameter marker in an SQL statement. When the statement is executed, the content of the variable is sent to the database server.

Syntax

`rc = RDBSetStr(ID: FieldNum: Value)`

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	Parameter marker number, ordered sequentially left to right, starting at 1.
Char(32767) Varying	Value	Input	The value to use in the corresponding parameter marker.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Set the value of the third parameter as a string with a value of VINET  
C      Eval      rc = RdbSetStr(Id: 3: 'VINET')
```

RDBSetDate (Set a timestamp field to a parameter marker)

Purpose

RDBSetDate associates a timestamp application variable or constant value to a parameter marker in an SQL statement. When the statement is executed, the content of the variable is sent to the database server.

Syntax

`rc = RDBSetDate(ID: FieldNum: Value)`

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	Parameter marker number, ordered sequentially left to right, starting at 1.
Timestamp	Value	Input	The value to use in the corresponding parameter marker.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
* Set the value of the first parameter as a Date
C      Eval      rc = RdbSetDate(Id: 1: ShipDate)
```

RDBSetNull (Set a NULL value to a parameter marker)

Purpose

RDBSetNull associates a NULL indicator to a parameter marker in an SQL statement. When the statement is executed, the database server field will be set to NULL.

Syntax

```
rc = RDBSetNull(ID: FieldNum: FieldType)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	Parameter marker number, ordered sequentially left to right, starting at 1.
Int(10)	FieldType	Input	The type of field being set to NULL by this function. Valid types are: Rdb_Array Rdb_Boolean Rdb_Char Rdb_Clob Rdb_Date Rdb_Decimal Rdb_Double Rdb_Float Rdb_Integer Rdb_Null Rdb_Numeric Rdb_Real Rdb_SmallInt Rdb_Time Rdb_TimeStamp Rdb_VarChar
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CL0N01Factor1+++++Opcode&ExtExtended-factor2+++++
* Set the value of the first parameter as a NULL
C      Eval      rc = RdbSetNull(Id: 1: Rdb Char)
```

RDBSetCommit(Set commitment control)

Purpose

RDBSetCommit set the automatic commitment control value. By default it is set to on, meaning all transaction are automatically committed. Setting this to off will for the need to either commit or rollback any transactions that are performed.

Syntax

```
rc = RDBSetCommit(ID: AutoCommit)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Boolean	AutoCommit	Input	Tells the remote DB engine if AutoCommit is true or false. The default is true.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++Opcode&ExtExtended-factor2+++++
* Set the commitment control status.
C      Eval      rc = RdbSetCommit(Id: *Off)
```

RDBCommit(Commit all transactions)

Purpose

RDBCommit commits all transactions that have been performed since that last commit or rollback. Transaction commit only applies to the transactions issued for the current ID. Closing the ID without a commit will rollback the transactions.

Syntax

```
rc = RDBCommit(ID)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Commit the transactions.  
C      Eval      rc = RdbCommit(Id)
```


RDBRollback(Rollback all transactions)

Purpose

RDBRollback reverses all transactions that have been performed since that last commit or rollback. Transaction rollback only applies to the transactions issued for the current ID. Closing the ID without a commit will rollback the transactions.

Syntax

`rc = RDBRollback (ID)`

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++
* Rollback the tranactions.
C      Eval      rc = RdbRollback(Id)
```

RDBAddRec(Add a record to the remote database)

Purpose

RDBAddRec will add a record to the remote database, based on a previously executed SELECT statement. The record structure **must** match the field definitions from the RDBFIELDS command. An external datastructure can be created for this process by issuing the RDBIMPORT command to an output file and using that file as a datastructure.

The fields selected by the SELECT statement **must** match the datastructure.

Syntax

rc = RDBAddRec(ID: Record)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(*)	Record	Input	A datastructure that represents the record to be written. The structure must match the field definitions from the RDBFIELDS command.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
D @Orders E DS ExtName(Orders) Inz
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
*   After the connection is established, execute a select statement.
C   Eval          rc = RdbExec IID:
      `Select * From [dbo].[Orders]`
*   Add a record to the remote file
      Eval          rc = RdbAddRec(id: @Orders)
```

RDBUpdRec(Update a record in the remote database)

Purpose

RDBUpdRec will update a last record read in the remote database, based on a previously executed SELECT statement. The record structure **must** match the field definitions from the RDBFIELDS command. An external datastructure can be created for this process by issuing the RDBIMPORT command to an output file and using that file as a datastructure.

The fields selected by the SELECT statement **must** match the datastructure.

Syntax

rc = RDBUpdRec(ID: Record)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(*)	Record	Input	A datastructure that represents the record to be written. The structure must match the field definitions from the RDBFIELDS command.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
D @Orders E DS ExtName(Orders) Inz
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++

* After the connection is established, execute a select statement.
C      Eval      rc = RdbExec(Id:
C          'Select * from [dbo].[Orders]')

* Fetch the next record of the result set that was generated with the
* previously executed statement.
C      Eval      rc = RdbFetchNxt(Id)

* Update the current record.
C      Eval      rc = RdbUpdRec(Id: @Orders)
```

RDBDelRec(Delete a record in the remote database)

Purpose

RDBDelRec will delete the last record read in the remote database, based on a previously executed SELECT statement.

Syntax

```
rc = RDBDelRec(ID)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++
* After the connection is established, execute a select statement.
C      Eval      rc = RdbExec(Id:
C      'Select * from [dbo].[Orders]')
* Fetch the next record of the result set that was generated with the
* previously executed statement.
C      Eval      rc = RdbFetchNxt(Id)
* Delete the current record.
C      Eval      rc = RdbDelRec(Id)
```

RDBNextSet(Move the cursor to the next result set)

Purpose

RDBNextSet will move the cursor to the next result set of a multiple result set call. If a second result set does not exist, an error will be returned. Once the cursor has been moved, the previous result set can not be accessed again.

Syntax

```
rc = RDBNextSet(ID)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++Opcode&ExtExtended-factor2+++++

* After the connection is established, execute a select statement.
C      Eval      rc = RdbExec(Id:
C          '{ Call MultiResultSet() }')

* Fetch the next record of the result set that was generated with the
* previously executed statement.
C      Eval      rc = RdbFetchNxt(Id)

* Move to the next result set
C      Eval      rc = RdbNextSet(Id)

* Fetch the next record of the result set that the cursor was just
* moved to.
C      Eval      rc = RdbFetchNxt(Id)
```

RDBStoredProc (Create a SQL statement to execute a stored procedure on the remote server)

Purpose

RDBStoredProc will execute a stored procedure on the remote server. The syntax for the statement must be valid on the remote server.

The SQL statement string may contain parameter markers. A parameter marker is represented by a “?” character, and indicates a position in the statement where the value of an application variable is to be substituted, when RDBPrepExec() is called. RDBSetStr(), RDBSetDate(), and RDBSetNum() are used to associate a application variable or constant value to each parameter marker.

RDB Connect() must be called before calling this function.

If a previous statement has been executed for this connection, RDBFreeStmt() must be called to close the cursor, before calling RDBPrepStmt ().

Syntax

rc = RDBStoredProc (ID: Statement)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Char(32767) Varying	Statement	Input	The stored procedure to be ran on the remote server.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++
*
* After the connection is established, prepare the stored
* procedure.
C      Eval      rc = RdbStoredProc (Id:
C      '{ Call CreateFullName(?,?,?) }')
```

RDBRegOutput (Register an output parameter of a stored procedure)

Purpose

RDBRegOutput register a parameter with an output marker. The function is only valid when RDBStoredProc has been used.

Syntax

```
rc = RDBRegOutput(ID: FieldNum: FieldType: FldScale)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number of the parameter that was used by a previously executed RDBStoredProc.
Int(10)	FieldType	Input	The type of field being registered by this function. Valid types are: Rdb_Array Rdb_Boolean Rdb_Char Rdb_Clob Rdb_Date Rdb_Decimal Rdb_Double Rdb_Float Rdb_Integer Rdb_Null Rdb_Numeric Rdb_Real Rdb_SmallInt Rdb_Time Rdb_TimeStamp Rdb_VarChar
Int(10)	FieldScale	Input	The number of decimal places to be returned by the stored procedure. This number must be zero or more.
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Set the value of the third parameter as a output field of type  
* character with a scale of 0.  
C      Eval      rc = RdbRegOutput(Id: 3: Rdb_VarChar: 0)
```

RDBGetParmNum (Get a numeric field from a stored procedure parameter)

Purpose

RDBGetParmNum retrieves the data from a numeric field in the stored procedure call. The function is only valid when RDBStoredProc has been used for an execution and a RDBRegOutput has been set for the requested field.

Syntax

number = RDBGetParmNum(ID: FieldNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number of the parameter that was used by a previously executed RDBStoredProc.
Number(30,9)	number	Output	The value of the field in the corresponding result set is returned. Zero is returned upon failure.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Fetch the first field of the parameters and return it as a number.  
C      Eval      Field1 = RdbGetParmNum(Id:1)
```


RDBGetParmStr (Get a character field from a stored procedure parameter)

Purpose

RDBGetParmStr retrieves the data from a numeric field in the stored procedure call. The function is only valid when RDBStoredProc has been used for an execution and a RDBRegOutput has been set for the requested field.

Syntax

String = RDBGetParmStr(ID: FieldNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number of the parameter that was used by a previously executed RDBStoredProc.
Char(32767) Varying	String	Output	The value of the field in the corresponding result set is returned. Blank is returned upon failure.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Fetch the first field of the parameters and return it as a character  
* field.  
C      Eval      Field1 = RdbGetParmStr(Id:1)
```

RDBGetParmDate (Get a date/time/timestamp field from a stored procedure parameter)

Purpose

RDBGetParmDate retrieves the data from a date/time/timestamp field in the stored procedure call. The function is only valid when RDBStoredProc has been used for an execution and a RDBRegOutput has been set for the requested field.

Syntax

Timestamp = RDBGetParmDate(ID: FieldNum)

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	The field number of the parameter that was used by a previously executed RDBStoredProc.
Timestamp	Timestamp	Output	The value of the field in the corresponding result set is returned. An initialized timestamp is returned upon failure.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++
* Fetch the first field of the parameters and return it as a timestamp
* field.
C      Eval      Field1 = RdbGetParmDate(Id:1)
```

****RDBCrtTable(Create a table in any Database configured in RDBCFCG).***

Purpose

RDBCrtTable is used to export/create IBMi iSeries tables into any database configured in RDBCFCG.

Syntax

rc = RDBCrtTable(id: vCreateTableStmt)

Notes: Do not include a 'CREATE TABLE' in the prepared statement, the API adds it automatically.

Function Arguments

Data Type	Argument	Use	Description
INT(10)	ID	Input	The ID that was returned from the RDBConnect Statement.
CHAR(4096)	CreateTable command	Input	This parameter must contain the correct syntax for the appropriate database where the table is being created.

Examples

```
//Prepare Statement
vCreateTableStmt = 'PERSON.TESTZIP (CITY CHAR(25), STATE CHAR(25), TEST_OPEN
CHAR(8), ROWID CHAR(10)) '

// Create Table
rc = RdbCrtTable(id: vCreateTableStmt );
```

****RDBSetIsoDate(Set ISO Date in result set)***

Purpose

RDBSetIsoDate sets the date under ISO Format to transfer from IBMi to a compatible remote database. The function is only valid when RDBPrepStmt has been previously executed.

Syntax

```
rc = RDBSetIsoDate(ID: FieldNum: ISODATE)
```

Function Arguments

Data Type	Argument	Use	Description
INT(10)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10)	FieldNum	Input	The field number of the parameter that was used by a previously executed RDBPrepStmt.
DATE (10)	Date in ISO Format (2014-01-01)	Input	The value of the field in the corresponding result set.

Examples

```
// Set the ISO Date value  
rc = RdbSetIsoDate(id: iQ: isoDate);
```

****RDBSetIsoTime(Set ISO Time in result set)***

Purpose

RDBSetIsoTime sets the time under ISO Format to transfer from IBMi to a compatible remote database. The function is only valid when RDBPrepStmt has been previously executed.

Syntax

```
rc = RDBSetIsoTime(ID: FieldNumber: TIME)
```

Function Arguments

Data Type	Argument	Use	Description
INT(10)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10)	FieldNum	Input	The field number of the parameter that was used by a previously executed RDBPrepStmt.
TIME (8)	Time in ISO Format (10.59.59)	Input	The value of the field in the corresponding result set.

Examples

```
// Set the ISO Time value  
rc = RdbSetIsoTime(id: iQ: IsoTime);
```

****RDBSetCharStr(Set characters stream after executing RDBPrepStmt)***

Purpose

RDBSetCharStr(Set character stream after a prepared statement. This allows up to 32767 characters.)

Syntax

```
rc = RDBSetCharStr(ID: FieldNum: CharacterVariable)
```

Function Arguments

Data Type	Argument	Use	Description
INT(10,0)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10,0)	FieldNum	Input	Prepared statement placeholder sequence number
CHAR(32767)	Character variable	Input	Character variable containing results

Examples

```
// Get list of Catalogs in remote database  
rc = RDBSetCharStr(id: 1: vCharStream);
```

****RDBGetCharStr(Get characters stream after executing RDBPrepExec)***

Purpose

RDBGetCharStr(Get character stream after a prepared statement. This allows up to 32767 characters to be retrieved from remote database.)

Syntax

```
vRetVal = RDBGetCharStr(ID: FieldNum)
```

Function Arguments

Data Type	Argument	Use	Description
INT(10,0)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10,0)	FieldNum	Input	Prepared statement placeholder sequence number
CHAR(32767)	Character variable	Output	Character variable receiving results

Examples

```
// Get list of Catalogs in remote database  
vRetVal = RDBGetCharStr(id: 1);
```

****RDBGetIsoTime(Get ISO Time in result set)***

Purpose

RDBGetIsoTime gets the time value under ISO Format from a compatible remote database to IBMi. The function is only valid when RDBPrepStmt or RDBExec has been previously executed.

RPGLE Definition

```
D ISOTime S T
```

Syntax

```
ISOTime = RDBGetIsoTime(ID: FieldNumber: TIME)
```

Function Arguments

Data Type	Argument	Use	Description
TIME(8)	ISOTime	Output	The ISO Time field returned from the RDBConnect Statement.
INT(10,0)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10)	FieldNumber	Input	The Field number from remote database table used in previous RDBPrepStmt or RDBExec

Examples

```
// Set the ISO Time value  
ISOTime = RdbGetIsoTime(id: FieldNumber);
```


****RDBGetIsoDate(Get ISO Date in result set)***

Purpose

RDBGetIsoDate gets the date value under ISO Format from a compatible remote database to IBMi. The function is only valid when RDBPrepStmt or RDBExec has been previously executed.

RPGLE Definition

```
D ISODate S D
```

Syntax

```
ISODate = RDBGetIsoDate(ID: FieldNumber)
```

Function Arguments

Data Type	Argument	Use	Description
DATE(10)	ISO Date	Output	The ISO Date field returned from the RDBConnect Statement.
INT(10,0)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10)	FieldNumber	Input	The Field number from remote database table used in previous RDBPrepStmt or RDBExec

Examples

```
// Set the ISO Date value  
ISODate = RdbGetIsoDate(id: FieldNumber);
```

****RDBAsclsoTime(Get ISO Time in result set by Field Name)***

Purpose

RDBAscIsoTime gets the time value under ISO Format from a compatible remote database to IBMi. The function is only valid when RDBPrepStmt or RDBExec has been previously executed.

RPGLE Definition

```
D ISOTime S T
```

Syntax

```
ISOTime = RDBAscIsoTime(ID: FieldName)
```

Function Arguments

Data Type	Argument	Use	Description
TIME(8)	ISOTime	Output	The ISO Time field returned from the RDBConnect Statement.
INT(10,0)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10)	FieldName	Input	The Field name /Column name from remote database table used in previous RDBPrepStmt or RDBExec

Examples

```
// Set the ISO Time value  
ISOTime = RdbAscIsoTime(id: FieldName);
```

***RDBAscIsoDate(Get ISO Date in result set by Field Name)**

Purpose

RDBAscIsoDate gets the date value in ISO Format from a compatible remote database to IBMi. The function is only valid when RDBPrepStmt or RDBExec has been previously executed.

RPGLE Definition

```
D ISODate S D
```

Syntax

```
ISODate = RDBAscIsoDate(ID: FieldName)
```

Function Arguments

Data Type	Argument	Use	Description
DATE(10)	ISODate	Output	The ISO Date field returned from the RDBConnect Statement.
INT(10,0)	ID	Input	The ID that was returned from the RDBConnect Statement.
INT(10)	FieldName	Input	The Field name /Column name from remote database table used in previous RDBPrepStmt or RDBExec

Examples

```
// Set the ISO Date value  
ISODate = RdbAscIsoDate(id: FieldName);
```

RDBSetInt (Set a Integer field to a parameter marker)

Purpose

RDBSetInt associates an integer application variable or constant value to a parameter marker in an SQL statement. When the statement is executed, the content of the variable is sent to the database server.

Syntax

```
rc = RDBSetInt(ID: FieldNum: NumValue)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	Parameter marker number, ordered sequentially left to right, starting at 1.
Number(10)	NumValue	Input	The value to use in the corresponding parameter marker. This parameter can take up to a max of 10 numbers
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1+++++++Opcode&ExtExtended-factor2+++++++
* Set the value of the third parameter as a numeric value without decimal
position up to a max of 10 numbers.
/Free
    rc = RdbSetInt(Id: 2: 5);
/End-Free
```

RDBSetNegInt (Set a negative Integer field to a parameter marker)

Purpose

RDBSetNegInt associates a negative integer application variable or constant value to a parameter marker in an SQL statement. When the statement is executed, the content of the variable is sent to the database server.

Syntax

```
rc = RDBSetNegInt(ID: FieldNum: NumValue)
```

Function Arguments

Data Type	Argument	Use	Description
Int(10)	ID	Input	The ID that was returned from the RDB Connect statement.
Int(10)	FieldNum	Input	Parameter marker number, ordered sequentially left to right, starting at 1.
Number(10)	NumValue	Input	The value to use in the corresponding parameter marker. This parameter can take up to a max of 10 numbers including a negative sign
Int(10)	rc	Output	Zero is returned from a successful execution. Negative one (-1) is returned from a failed execution.

Examples

```
CLON01Factor1++++++Opcode&ExtExtended-factor2++++++  
* Set the value of the third parameter as a negative numeric value without  
decimal position up to a max of 10 numbers.  
/Free  
    rc = RdbSetNegInt(Id: 2: -516);  
/End-Free
```